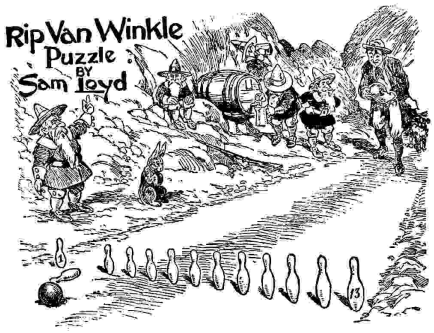
Problem C: Rip Van Winkle Puzzle

Rip Van Winkle is playing an old variant of the Dutch game *Kugelspiel*, where two players take turns to knock down pins using a ball. The pins are placed in a row, and the players stand close enough to make it very easy to knock one or two adjacent pins at a time. The players have to knock down at least one pin each turn, and the winner is the player who knocks down the last pin.



Rip Van Winkle playing Kugelspiel

In the figure you can see Rip Van Winkle going against a man of the mountain in a game that started with 13 pins. The little man played first and knocked down pin number two. It's Rip's turn now and he can choose among 22 different moves —he can knock down any single remaining pin (12 moves), or he can choose any pair of adjacent pins (10 moves).

Assuming that both players play optimally from this point until the end of the game, which moves can Rip choose for the next turn that will lead him to a victory?

Consider an arbitrary setup of this game that starts with **N** pins (enumerated $1, 2, 3, \ldots, N$). Some of the pins may be down already from previous turns, but there will always be at least one pin standing. Identify all winning moves for the player who moves next.

Input

Input starts with a positive integer **T**, that denotes the number of test cases.

The first line of each test case contains the number **N**. The second line begins with an integer **K** that denotes the number of pins that have been knocked down already, and is followed by K integers in strictly ascending order, that indicate the pins that are down.

$T \leq 3000$; $1 \leq N \leq 100$; $K < N$

Output

For each test case, print the case number followed by the number of winning moves.

Then print the moves, one per line using the following format:

- If the move is to knock down pin number **i**, then print **(i)**.
- If the move is to knock down two adjacent pins **i** and **j** (with $j = i + 1$) then print **(i, j)**.

Print all single–pin moves first in ascending order, and then print all two–pin moves, sorted by their lowest pin.

Sample Input	Output for Sample Input
3 13 1 2 23 8 1 3 7 8 10 13 14 21 7 0	Case 1: 2 (6) (10) Case 2: 0 Case 3: 3 (4) (2, 3) (5, 6)